Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2007 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2007

Facilitating Design Knowledge Management by Tailoring Software Patterns to Organizational Roles

Riikka Ahlgren
University of Jyvaskyla

FACILITATING DESIGN KNOWLEDGE MANAGEMENT BY TAILORING SOFTWARE PATTERNS TO ORGANIZATIONAL ROLES

Riikka Ahlgren
Information Technology Research Institute, University of Jyväskylä
P.O. Box 35, FIN-40014 University of Jyväskylä, Finland
ahlgren@titu.jyu.fi

Abstract

This paper considers knowledge management in software development from organizational roles' viewpoint. The focus is on software patterns, which are seen as means to facilitate the design knowledge management. In this paper, patterns are promoted by tailoring their presentation according to the specific needs of each role. Roles' needs are justified by a case study made in a Finnish software organization. Pattern presentation formats are analyzed considering different skills of roles. The analysis reveals that different roles can best utilize different parts of pattern descriptions. As a result, a tentative model is built where relationships between roles and pattern parts are depicted. The result can be used to promote patterns as means for managing the design knowledge in organizations.

Keywords: *Software patterns, Knowledge management, Organizational roles*

Introduction

Software organizations possess a vast amount of knowledge about their business and the technologies they use. Design knowledge, together with business knowledge and technical skills, form the core of a software organization's knowledge (Stein 1995). Several knowledge management activities have been identified, which support the software development process (Rus et al. 2002). Suggested activities include for example document management, competence management, and software reuse (Rus et al. 2002). This paper utilizes and builds on the reuse perspective by considering patterns and their reuse as a knowledge management problem, as suggested by Highsmith (2003).

Contents of design knowledge are composed of design trade-offs between various software qualities, which are due to particular structural or behavioral aspects of the system (Gross et al. 2001). The term "design knowledge management" can be defined as management of explicit and tacit design knowledge (Kucza et al. 2001). The concept of reuse in turn, is merely focused on reusing explicit components (Kucza et al. 2001). Approaching the knowledge management with Nonaka's terms (Nonaka 1994), the focus of this research lay on the knowledge combination phase of the knowledge creation cycle. In this phase the design knowledge is shared between the roles who participate the development process.

Recent literature identifies problems related to knowledge sharing in software development (DeSouza 2003; Rus et al. 2002; Selfridge et al. 1992). The problems can be categorized as technological, organizational and individual problems (Rus et al. 2002). A general organizational problem is the lack of common language of developers and other roles (Buschmann et al. 1996; Walz et al. 1993), which is due to their different backgrounds, skills and interests. This problem becomes visible for example, when developers should present their work estimations in an understandable way to the project manager. Even if the project manager has a technical background, he or she does not necessarily have enough knowledge about current design solutions. Hence, a common vocabulary may be difficult to establish.

However, the management of design knowledge is more often focused on tools (DeSouza 2003; Gomes et al. 2006; Kucza et al. 2001) than on the people and process perspective (Kucza et al. 2001). Thus, to get a broader view of the people perspective in knowledge management, we focus on organizational roles and particularly on project managers. This focus is motivated by the organizational setting, where software is built, namely project business. Project managers are the key in deciding how much time the developers can use to each project task. Thus, to enable better management of design knowledge, and further to be able to allocate time for the pattern related activities, the project managers must be aware of the design issues at least in general level.

Software patterns have been suggested to facilitate management of design knowledge and software reuse, since they capture both the design decisions made but also the decision rationale in a compact, easily-available and structured format (Buschmann et al. 1996). Previously patterns have been adopted both in object-oriented environment but also in other domains and these demonstrate capture, transfer and management of design knowledge (May et al. 2003). Further, previous research provides empirical evidence on patterns' benefits regarding design quality and communication improvement inside the development team (Beck et al. 1996; Cline 1996).

However, the adoption of patterns in organizations is a complex issue. The impact of project managers (Cline 1996) and other organizational factors (Manns 2002) have been acknowledged in previous research. Manns (2002) has created guidelines for facilitating the pattern adoption in organizations. One of the guidelines proposes that the management should find an appropriate level of support for patterns, which will help the adoption. Another guideline suggests that the organization should find effective ways to make patterns visible throughout the organization and further, that an organization should help individuals understand pattern descriptions. Our research builds on these findings by suggesting that pattern presentation formats can be tailored according to the needs of different organizational roles. We propose that tailoring can lower the existing knowledge barriers (see Manns 2002), and thus make the patterns more understandable.

The research question addressed in this paper is *"How to demystify software patterns for the business-oriented roles of software development?*" The paper analyses pattern descriptions according to the skills needed to understand and learn the pattern descriptions. The aim is to enable different roles better to exploit the patterns, and thus to facilitate the design knowledge management during the software development. The tasks and the skills that are needed for various roles are considered, the presentation formats of patterns are analyzed, and the skills needed to understand different pattern parts are identified.

The paper is organized as follows. The following section describes the case setting and the research method. Software patterns as knowledge entities are presented after that, which is then followed by the organizational roles and their tasks. Pattern presentation formats are demonstrated and parts of the patterns are analyzed. Finally, a tentative framework is presented, which combines the roles and the different parts of patterns.

Case Description and Research Method

The research was motivated with a case study. The case organization is a large Finnish software company specializing in integrated ICT solutions and mobile products and services. The company stresses quality

and efficient working habits, thus processes, quality assurance and time management are effectively implemented. Software processes as well as related practices are comprehensively documented.

Design knowledge management has been one of the company's central improvement targets and during the past years several improvement efforts have been implemented. However, in daily development work the use of software patterns has merely been up to individual developers more than a common practice. As a result, the management of design knowledge has not considerably improved. Studying the patterns on free-time, as suggested by previous research (Cline 1996), neither improved the management of design knowledge.

In the case company, unsuccessful realization of some improvements was mainly due the insufficient organizational support. For example, the time management system did not recognize the design knowledge management activities as tasks to be tracked. Furthermore, templates for project planning did not support those activities in adequate manner, thus project managers did not allocate time to the particular tasks.

To make the management of design knowledge more efficient and common, and further to enable reuse of software artifacts, the company desired to raise the abstraction level of reuse and became interested in software patterns. A less detailed format of design information is known to facilitate its transfer (May et al. 2003) and hence, it was supposed to make the knowledge more understandable for more roles in the case company and further to increase the organizational support for design knowledge management.

However, introduction of patterns did not solve original problems: patterns' use was not supported by project management and by time allocation. Project managers and people in charge of time management system did not recognize patterns or their benefits. Thus, there was a need to convince project managers and others in non-technical roles of the advantages of patterns. This motivates the research reported in this paper.

Data collection for the case study was made during winter 2005/2006. The research method included semi-structured interviews, meeting discussions and an extensive document analysis. The analyzed documents included a quality handbook, process descriptions and various document templates. Semi-structured interview method was chosen to keep the interviews informal and conversational, and thus to encourage the interviewees to also talk about the difficulties encountered.

Designers, architects and managers were interviewed in three separate sessions. First, the quality manager, who also worked as a project manager, was interviewed in order to identify the company's objectives for the pattern promotion. Then an extensive documentation analysis was made to define the roles to be interviewed and to identify the skills needed in their tasks. The second and third interview was conducted to verify the findings of the documentation analysis, including the needs of the roles and their required skills. The interviews lasted 30 and 45 minutes, and there were eight programmers, designers or architects in both sessions. In total 12 persons were interviewed, for some people attended both sessions.

In addition to the interviews, notes were taken in three other meetings, where patterns were presented and discussed. In the first meeting there were 6 designers and the technology manager. The last two meetings were held with the company management. In addition to the technology manager of the case company, there attended four managers from other companies who also were interested in promoting patterns.

Patterns as Knowledge Entities

Software patterns are reusable knowledge entities, more concrete than components but less concrete than frameworks (Fayad et al. 1999). A classical definition for a pattern is "a three part rule, which expresses the relation between a certain context, problem and solution" (Alexander 1979). Patterns can be classified to idioms, design patterns and architectural patterns (Gamma et al. 1995) and are recognized tool to learn, document and to share experimental design knowledge (Buschmann et al. 1996; Gamma et al. 1995). The idea of software patterns as knowledge entities is supported by their close connection to organizational memory (Ahlgren et al. 2006). Further, the problems faced with introducing patterns are similar to the

knowledge management problems (May et al. 2003). These findings indicate that knowledge management practices can be helpful in managing patterns as well.

In software development the patterns are experienced as precise enough to preserve the domain-specific knowledge but still flexible enough allowing the systems' future modifications (Olson 1998). This two-fold abstractness is one of the patterns' key characteristics in organizational context, for it enables the patterns to be used for different purposes by different roles.

Different pattern presentation formats are suggested in literature (Alexander et al. 1977; Buschmann et al. 1996; Coplien 2000; Gamma et al. 1995) indicating that different formats can be useful also in practical software development: more extensive descriptions for technical roles and particular for novices, and brief overview for non-technical roles and experts, who already know the pattern but need support for their memory. In literature, however, patterns are mainly discussed from the developer's point of view, thus forgetting the different backgrounds, skills and needs of other roles in the software development team.

Various skills are required for understanding pattern descriptions. Traditionally patterns are targeted at people skilled with programming and UML-notation (Buschmann et al. 1996; Gamma et al. 1995). Although even a brilliant OO programmer may not understand OO design, a basic assumption is that the principles of object-oriented design are known (Holub 2004). This makes patterns burdensome to study by people unskilled in programming. However, as the previously described case study and literature indicates (Cline 1996), also other roles can benefit of patterns, for example project managers.

Different Roles and Their Languages

Organizations are composed of actors performing their roles. In order to have efficient software development teams, the teams are composed of roles with different skills (Curtis et al. 1988). The development projects typically include tasks where actors with non-technical skills are needed. In this paper, we will use the term non-technical role to indicate an actor whose primary tasks are business-oriented, not code-centered.

In order to ensure flexible and efficient software development, the communication between the roles must be fluent (Walz et al. 1993). Problems arise when roles have different viewpoints and vocabularies of the software under development. Software patterns are proposed to facilitate the creation of common language (Buschmann et al. 1996; Gamma et al. 1995), and thus to facilitate the knowledge transfer. Considering design knowledge management, communication between technical and non-technical roles is particularly needed to assure that enough time is allocated for the pattern related tasks. As we present in this paper, the approach to patterns may vary from role to role, according to their needs, interests and skills.

Considering software projects in the case company, we identified five key roles, which typically formed a development team. The roles are architect, designer, programmer, project manager and product manager. The three first are seen as technical roles and two latter as non-technical ones. In the figure 1 the development team is illustrated in the context of their task technicality.

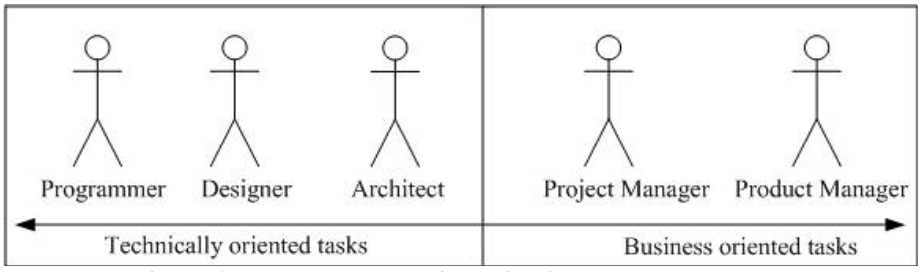

Figure 1. Roles and task orientation in development team.

The case company has several architects, who are responsible for the software architecture, including definitions of central design patterns. An architect can have overlapping responsibilities in several development teams. As a technical role, the architect often has the best knowledge of the possible system restrictions to the software. Architect is able to read and draw specifications using graphical notation.

Designers and programmers are often called as software engineers or developers. A designer has a responsibility of designing entire modules, and he specifies the detailed key technical issues of the source and test code. Programmer, in turn, creates the source code of the software. In the case company the practices set very few requirements on source code, hence an effective technical implementation is dependent on the developers' personal skills. Designer is able to read and draw graphical specifications and has programming skills. Programmer is able to read the given specifications and his responsibility is to implement the product with given instructions. Although in practice the roles of designer and programmer are often performed by one actor, we prefer to handle them as separate roles, in order to clarify the different tasks and responsibilities of these roles and to clarify the different skills that are needed in their tasks.

Project managers' tasks in the company are to create and update the project plan. Skills in resource management and risk management are central. He is familiar with the business domain and may have been a designer or an architect previously. Thus he can read graphical specifications but typically has no time for technical details or latest programming practices. In the case company, every product has a product manager with a business approach. The tasks include executing and controlling the planned product activities, which are implemented by several development teams. Technical issues can be rather uninteresting to this role and skills concerning for example reading UML notations can be limited.

The interests of the roles may vary greatly. Non-technical roles, meaning product and project managers, are mainly interested in if the product fills the requirements, how it is used in real life and when it can be delivered. The technical role, in turn, is interested for example in technologies that can be used and the trade-offs of each alternative design.

Differences in the roles' skills and their varying interests have a significant effect on roles' abilities and willingness to understand and exploit patterns. Hence, pattern presentation formats can be tailored to be more suitable for a wider audience, thus making the patterns more understandable and more useful for the non-technical roles as well.

Formats of Pattern Presentation

There are three pattern presentation formats commonly used in literature: Alexandrian, GoF and Buschmann's formats (Alexander et al. 1977; Buschmann et al. 1996; Gamma et al. 1995), illustrated in figure 2. Most pattern presentation formats include three basic parts: the problem, its context and the proposed solution, although some other groupings exist as well (Cechich et al. 1999; Coplien 2000; Soundarajan et al. 2004).

Alexandrian format:	GoF format:	Buschmann format:
Title	Pattern name and	Name
Picture	classification	Also known as
Introduction	Intent	Example
Headline	Also known as	Context
Body	Motivation	Solution
Solution	Applicability	Structure
Diagram	Structure	Dynamics
Related patterns	Participants	Implementation
	Collaborations	Example resolved
	Consequences	Variants
	Implementation	Known uses
	Sample Code	Consequences
	Known Uses	See also
	Related Patterns	

Figure 2. Common pattern presentation formats.

Regardless of the presentation format, pattern Name is the most important section (Coplien 2000). Names distinct patterns from each other and enable pattern comparisons. By name the whole pattern content can be discussed without explaining the details, thus pattern names can be a part of design vocabulary. One pattern can have several names, presented in Also known as -section.

The Motivation provides a practical example of patterns context, problem and solution. The purpose is to give an overview of the area that pattern concerns. It usually is followed by an explanatory text. An example of descriptive picture is illustrated in figure 3, where the Observer –pattern is described as a single picture. The text then describes how the information in different objects can be updated by using an observer to mediate the changes (Gamma et al. 1995).

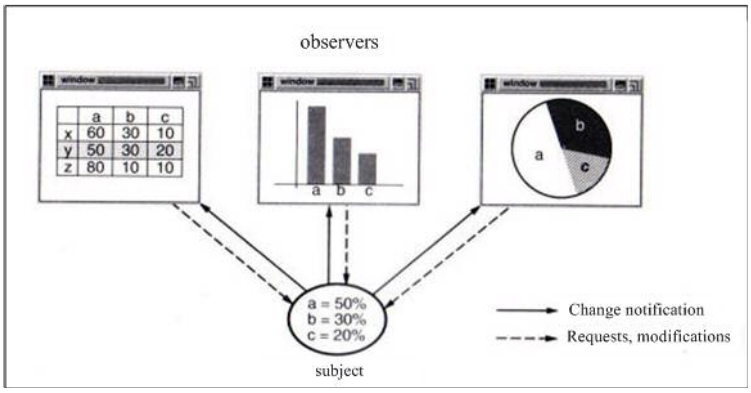

Figure 3. Observer -pattern in motivation –section (Gamma et al. 1995).

The Intent summarizes the general problem and introduces the solution in brief form, while the context of the problem addressed by the pattern is presented in the Context or Motivation sections. The context describes the factors that must be present for the pattern to work, for example programming language, size or scope. The Applicability describes situations where the pattern may be adopted successfully; in a same manner the problem declares the details in the problem area. A concise problem statement helps the problem solver to decide whether to keep on reading this particular pattern description (Coplien 2000). Participants are the classes or objects that play key roles in implementing the function that the pattern describes. The Structure, in turn, includes the collaborations of the participants and presents them in graphical notation, as illustrated in the following figure 4.

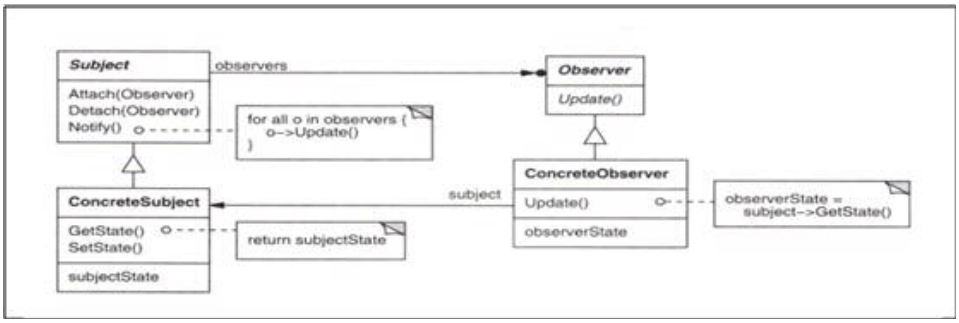

Figure 4. The structure of Observer -pattern illustrated with UML –notation (Gamma et al. 1995).

In the Dynamics or Collaborations sections, the participants' behavior is described in more detail, emphasizing the interactions of the pattern. Sequence charts are commonly used to describe the behavior and timely interactions of each object, as illustrated in the following figure 5.

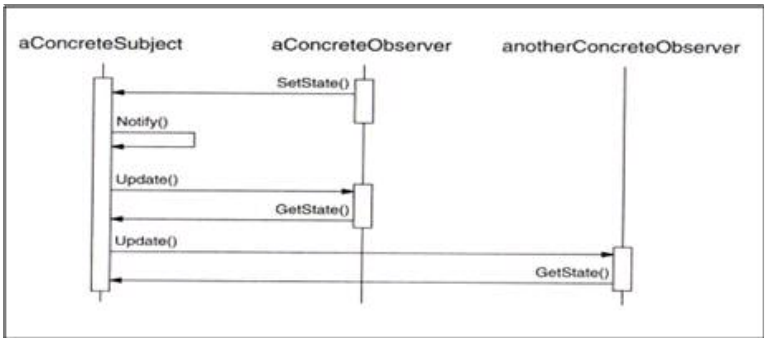

Figure 5. The dynamics of observer -pattern in collaboration –section (Gamma et al. 1995).

The Implementation section facilitates to implement the pattern in practice. Separate code examples are given in the Sample code or Example resolved sections. The Consequences section describes the benefits and trade-offs, which the pattern adoption brings along. The Known uses section, in turn, describes previous successful adoptions, thus facilitating the evaluation of the pattern appropriateness. The Related patterns and See also sections introduce other similar patterns, which can be used for the particular problem. Browsing through other possible patterns may provide information about the pattern context and applicability, and suggest suitable pattern combinations.

The GoF pattern presentation format is particularly intended to help users in creating solutions to problems (Beck et al. 1994). The users focus less on when to apply the pattern, and more on the actual structure and dynamics of the pattern itself. Thus, this pattern format is more descriptive rather than generative (Beck et al. 1994). Therefore, the GoF format is selected for analyzing the contents of the presentation format.

Analysis of Pattern Presentations

The purpose of each section of the pattern description is not always unambiguous. This is illustrated in the following figure 6 where the sections of GoF presentation format are depicted according to the Alexandrian three part rule, including context, problem and solution.

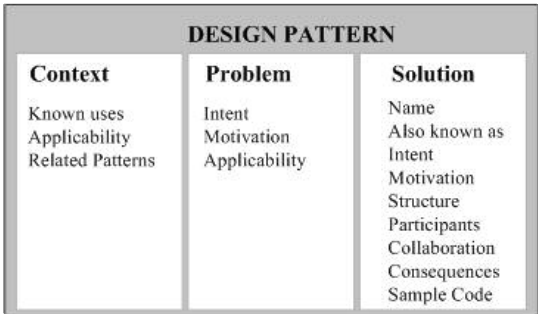

Figure 6. GoF presentation format as a three part rule.

The pattern sections Applicability, Intent and Motivation are present in more than one part, which enables logical connections between the parts. The Applicability section is used both in finding the right context of the pattern, as well as in identifying the similarity between the problem areas of the current situation and the one described in the pattern description. Correspondingly, the Intent section is used to identify the suitable problem area, and it also connects the problem area with the possible solution by summarizing the pattern's purpose. This connection is provided also by the Motivation section, but in a more structural way. Together the Applicability, Intent and Motivation sections provide a consistent path from the context to the solution, which is essential particularly when learning the pattern for the first time.

The Related patterns section is placed to the context part, for related patterns give the adopter hints about the environment where the pattern can be used. On the other hand, it could be placed to the solution part, for related patterns are links to the pattern system, thus facilitating the navigation in the pattern system and helping to understand the entity of patterns. These links between the patterns describe specific pattern combinations which may serve as a solution. The pattern Name is even more problematic than the Related patterns, for there are no rules of how to give a name to a new (in-house) pattern, since the main purpose of a pattern name is to be informative and descriptive.

Pattern Parts in Design Knowledge Management

The analysis of the case study reflected that reuse was a knowledge management problem indeed, which could not be solved only by raising the abstraction level. This is indicated also by Highsmith (2003). Considering the role of project managers, the case findings indicated that design knowledge management was not supported by the development processes. Similar findings are reflected also by Komi-Sirviö et al. (2002).

Considering pattern presentation from the design knowledge management perspective, the likely use of patterns depends on the roles in the development team. The relationships between the roles and the pattern use are illustrated in figure 8. The intensity of the arrow indicates the significance of different pattern parts to each role. The intention is to present the most essential patterns parts regarding to the tasks of the roles, without excluding others.

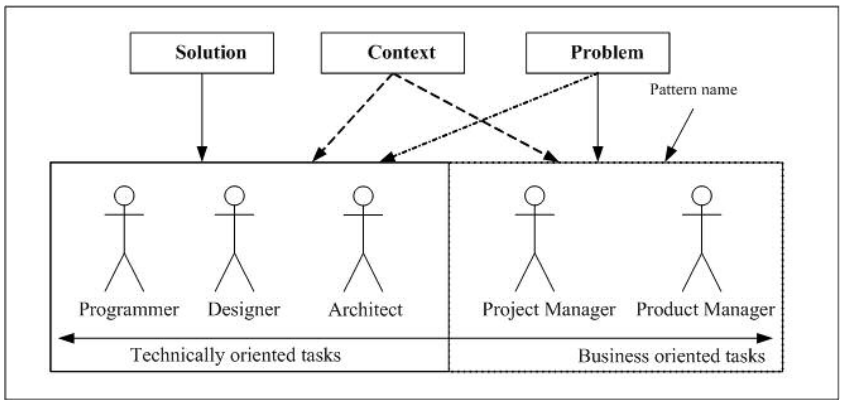

Figure 7. Essential pattern parts and development team roles

Some of the sections in pattern descriptions rely heavily on code, thus programming skills in appropriate language are needed for learning and adopting the pattern in a design. Particularly the sections, which are meant to facilitate the pattern adoption and making the right modifications, for example sections "Implementation" and "Sample code", often heavily employ source code in a specific programming language.

Many designers and programmers prefer software-centric visual aids such as class models and interaction graphs (Schmidt et al. 1996). Therefore, many pattern description formats use popular notations (such as Booch models and OMT) to express the structure and dynamic behavior concisely (Schmidt et al. 1996). This highlights the importance of graphical designing language knowledge.

The technically oriented roles need to understand the structure, participants and the collaborations most profoundly, and thus the solution part seems to be the most valuable in their work. The solution part is often described with programming language and graphical notation, which are familiar and useful for these roles. For this particular reason, the business oriented roles may not be able to interpret the solution part. It should be mentioned that, while the context and the problem parts are of little (if any) interest to the programmer role, they are important for the designer and especially for the architect.

The non-technical roles need merely the problem part and the pattern name in their work. The Motivation section gives managers a good overview of the proposed pattern, without going too much into details. Additionally, the Motivation is often described with plain text and explaining pictures (Gamma et al. 1995), instead of graphical notations or programming languages, which facilitates the understanding.

It should be noted that, considering the roles in general level, the managers often are not interested on problems, but on solutions (Kolb 1996). However, in the case of patterns the aim is not to make the managers learn the pattern contents in detail, but to facilitate their understanding and communication about the design issues. The problem –part of pattern descriptions gives the best overview of the pattern, and therefore it seems to best suit for the non-technical roles. Furthermore, as indicated in the research of Curtis and others (Curtis et al. 1988), gaining better understanding of design issues facilitates the project managers' actual task as well, namely estimating the needed time for development tasks.

As well technical as non-technical roles are able to use the context part, in order to support their thinking and decision making. Particularly the motivation section can be exploited by all roles, for it explains the pattern core in layman's terms, thus being both informative and abstract enough.

Thus, when creating an in-house pattern presentation format, the skills of the different roles should be acknowledged, and different parts of patterns should be described in an understandable way, not only for the technical people but also to the business oriented roles. However, the pattern presentation format used in a company does not have to follow the formats presented in the literature; rather, patterns can be formatted according to the organization's own needs and practices. It is up to the organization to decide what are seen important and which not, although some basic information, pattern name for example, should always be included, because of its functionality.

Concluding Remarks

In this paper the management of design knowledge is studied through software pattern presentation formats. The patterns are a tool to disseminate the design knowledge inside a company and a tool to facilitate the communication during software development. Furthermore, pattern presentation formats can enable different readers to manage and exploit the knowledge captured in pattern format. Despite of the fact that the patterns described and analyzed in this paper were design patterns, the principle of tailoring the pattern presentation formats can be adapted to the architectural patterns as well. This is enabled by the similar composition of different levels of software patterns (Buschmann 1996)

In software development, communication problems are likely to occur when managers have different viewpoint and vocabulary than the actual users of patterns, namely programmers, designers and architects. In order to facilitate the design knowledge management and to facilitate the communication of design issues, patterns should be understood by different roles, despite the differences of their skills. Meanwhile, pattern catalogues usually present patterns in a single format, not adjusted to different roles. Therefore, the proposed approach emphasizes dialogue between the individuals, as encouraged by DeSouza (2003).

A case study was made to motivate the research. The case revealed organizational factors related to patterns and their use, which should be acknowledged when patterns are introduced in an organization. Based on the case, the research question for this paper was stated as *"How to demystify software patterns for the business-oriented roles of software development?"* To address this question, pattern presentation was analyzed considering the possessed skills of different roles in software development. It was concluded that different roles can utilize different parts of pattern descriptions. Non-technical roles get best understanding of the design issues by using the problem part of the pattern description, while technical roles can better exploit the solution part.

The empirical validation of the results is an ongoing process. The results of this research have been presented to the case organization and consequently, a few improvements have been initiated. However, the details of the initiated improvements are considered to be business sensitive information, thus they are not

yet public. Since any changes to the organizational processes are slow to implement, the consequences of the initiatives will be a topic for future research.

The role of a mentor can not be forgotten when promoting patterns in organizations (Beck et al. 1996). However, this role is hard to fulfill, unless the developers have no time to learn the patterns. We suggest that project managers can facilitate mentors' work by acknowledging the patterns and further by allocating time for the developers to study them. This suggestion is consistent with the previous research findings (Manns 2002).

Taken together, patterns' use needs organizational support as any other knowledge management activity. The support is hard to grant and implement, if the supported matter or its benefits are too burdensome to understand by the roles that make the actual decisions. Hence, to ensure patterns' organizational support, their use must be motivated for project managers and other managers as well.

Therefore, when introducing patterns for the managers, problem part can be emphasized to ensure the best understanding of the contents. Furthermore, a pattern presentation format employed in an organization can be adapted to the needs of a particular role to ensure a wider audience for the patterns. Technical tools can be utilized to automatically produce different views of a same pattern.

Acknowledgments

The research work presented in this paper was conducted in MODPA research project [http://www.titu.jyu.fi/modpa] at the Information Technology Research Institute, University of Jyväskylä. MODPA project was financially supported by the National Technology Agency of Finland (TEKES) and industrial partners Nokia, SysOpenDigia, SESCA Technologies, Tieturi, Metso Paper and Trusteq. The author wishes to thank the MODPA project group and Samuli Pekkola for their comments and support.

References

Ahlgren, R., Markkula, J. "Design Patterns and Organisational Memory in Mobile Application Development", PROFES 2005, Springer-Verlag, Oulu, Finland, 2005, pp. 143-156.

Alexander, C. The Timeless Way of Building, New York: Oxford University Press, 1979.

Alexander, C., Ishikawa, S., Silverstein, M., Jakobson, M., Fiksdahl-King, I., and Angel, S. A Pattern Language (Volume 2). New York: Oxford University press, 1977.

Beck, K., Crocker, R., Meszaros, G., Coplien, J.O., Dominick, L., Paulisch, F., and Vlissides, J. " Industrial experience with design patterns," 18th International Conference on Software Engineering, IEEE, Berlin, Germany, 1996, pp. 103-114.

Beck, K., and Johnson, R. "Patterns Generate Architectures," 8th European Conference of Object-Oriented Programming, Bologna, Italy, 1994.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. Pattern-oriented software architecture - A system of patterns, Chicester, West Sussex, England: John Wiley & Sons ltd., 1996.

Cechich, A., and Moore, R. "A Formal specification of GoF Design Patterns," The United Nations University, International Institute for Software Technology, 1999.

Cline, M.P. "The pros and cons of adopting and applying design patterns in the real world," Communications of the ACM (39:10), 1996, pp. 47-49.

Coplien, J. Software Patterns, New York: SIGS Books & Multimedia, 2000.

Curtis, B., Krasner, H., and Iscoe, N. "A field study of the software design process for large systems," Communications of the ACM (31:11), 1988, pp. 1268-1287.

DeSouza, K.C. "Barriers to Effective Use of Knowledge Management Systems in Software Engineering," Communications of the ACM (46:1), 2003, pp. 99-101.

Fayad, M., Schmidt, D.C., and Johnson, R. Building Application Frameworks, New York: John Wiley & Sons Inc., 1999.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of reusable object oriented software, Boston, USA: Addison-Wesley, 1995.

Gomes, P., and Leitão, A. (eds.) A Tool for Management and Reuse of Software Design Knowledge, Berlin -Heidelberg: Springer Verlag, 2006.

Gross, D., and Yu, E. "From Non-Functional Requirements to Design through Patterns," Requirements Engineering (6), 2001.

Holub, A. Holub on Patterns: Learning design patterns by looking at code, USA: aPress, 2004.

Kolb, D.A. "Management and the learning process," in: How Organizations Learn, K. Starkey (ed.): International Thomson Publishing Company, London, 1996, pp. 270-287.

Komi-Sirviö, S., Mäntyniemi, A., and Seppänen, V. "Toward A Practical Solution for Capturing knowledge for Software Projects," IEEE Software (May/June), 2002, pp. 60-62.

Kucza, T., Nättinen, M. and Parviainen, P., "Improving knowledge management in software reuse process," PROFES, Springer-Verlag, Kaiserslautern, Germany, 2001, pp. 141-152.

Manns, M. L. “An investigation into factors affecting the adoption and diffusion of software patterns in industry”, De Montfort University, Leicester, United Kingdom, 2002.

May, D., Taylor, P. “Knowledge management with patterns,” Communications of the ACM (46:7), 2003.

Nonaka, I. "A Dynamic Theory of Organizational Knowledge Creation," Organization science (5:1), 1994.

Olson, D. "FAQ from Bruce Anderson's group at ChiliPLoP'98," Wickenburg AZ, 17-20 March 1998, 1998.

Rus, I., and Lindvall, M. "Knowledge Management in Software Engineering," IEEE Software (19:3), 2002, pp. 26-38.

Schmidt, D.C., Fayad, M., Johnson, R.E., and Guest Editors "Software Patterns," Communications of the ACM (39:10), 1996.

Selfridge, P.G., Terveen, L.G., and Long, M.D. "Managing design knowledge to provide assistance to large-scale software development," Knowledge-Based Software Engineering Conference, IEEE, McLean, VA, USA, 1992, pp. 163-170.

Soundarajan, N., and Hallstrom, J.O. "Responsibilities and Rewards: Specifying Design Patterns," 26th International Conference on Software Engineering, IEEE Computer Society, Edinburgh, Scotland, 2004.

Stein, E.W. "Organizational Memory: Review of Concepts and Recommendations for Management," International Journal of Information Management (15:1), 1995, pp. 17-32.

Walz, D.B., Elam, J.J., and Curtis, B. "Inside Software Design Team: Knowledge Acquisition, Sharing and Integration," Communications of the ACM (36:10), 1993, pp. 63-77.